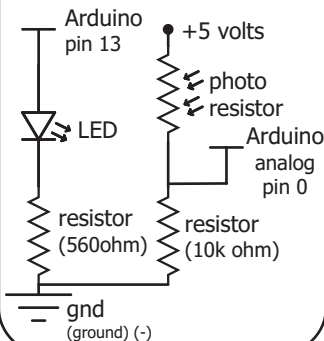


**WHAT WE'RE DOING:**

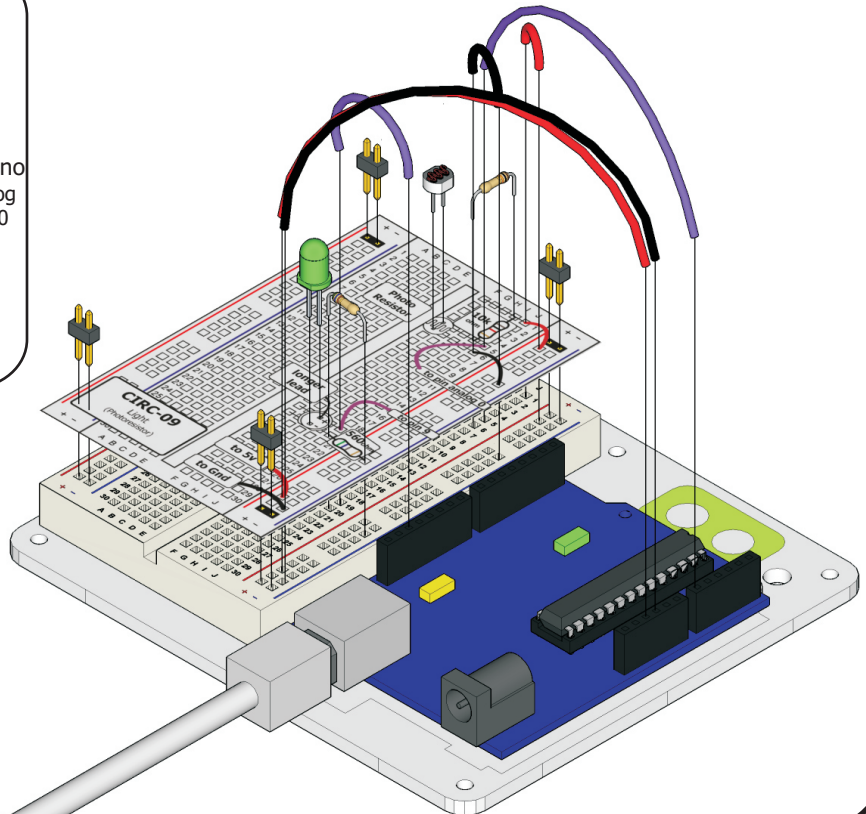
Whilst getting input from a potentiometer can be useful for human controlled experiments, what do we use when we want an environmentally controlled experiment? We use exactly the same principles but instead of a potentiometer (twist based resistance) we use a photo resistor (light based resistance). The Arduino cannot directly sense resistance (it senses voltage) so we set up a voltage divider (<http://ardx.org/VODI>). The exact voltage at the sensing pin is calculable, but for our purposes (just sensing relative light) we can experiment with the values and see what works for us. A low value will occur when the sensor is well lit while a high value will occur when it is in darkness.

**THE CIRCUIT:****Parts:****CIRC-09  
Breadboard Sheet  
x1****2 Pin Header  
x4****Photo-Resistor  
x1****Wire****10k Ohm Resistor  
Brown-Black-Orange  
x1****560 Ohm Resistor  
Green-Blue-Brown  
x1****Green LED  
x1****Schematic****The Internet****..:download:..**

breadboard layout sheet

<http://ardx.org/BBLS09>**..:view:..**

assembly video

<http://ardx.org/VIDE09>

**CODE** (no need to type everything in just click)**Download the Code from ( <http://ardx.org/CODE09> )**

(copy the text and paste it into an empty Arduino Sketch)

```

/*
 * A simple programme that will change the
 * intensity of an LED based on the amount of
 * light incident on the photo resistor.
 */

//PhotoResistor Pin
int lightPin = 0; //the analog pin the
                  //photoresistor is
                  //connected to
                  //the photoresistor is not
                  //calibrated to any units so
                  //this is simply a raw sensor
                  //value (relative light)

//LED Pin
int ledPin = 9; //the pin the LED is connected to
                //we are controlling brightness so
                //we use one of the PWM (pulse
                //width modulation pins)

void setup()
{
  pinMode(ledPin, OUTPUT); //sets the led pin to

```

```

//output
}
/*
 * loop() - this function will start after setup
 * finishes and then repeat
 */
void loop()
{
  int lightLevel = analogRead(lightPin); //Read the
                                          // lightlevel
  lightLevel = map(lightLevel, 0, 900, 0, 255);
              //adjust the value 0 to 900 to 0 to 255
  lightLevel = constrain(lightLevel, 0, 255);
              //make sure the value is between 0 and 255
  analogWrite(ledPin, lightLevel); //write the value
}

```

**NOT WORKING?** (3 things to try)**LED Remains Dark**

This is a mistake we continue to make time and time again, if only they could make an LED that worked both ways. Pull it up and give it a twist.

**It Isn't Responding to Changes in Light.**

Given that the spacing of the wires on the photo-resistor is not standard, it is easy to misplace it. Double check its in the right place.

**Still not quite working?**

You may be in a room which is either too bright or dark. Try turning the lights on or off to see if this helps. Or if you have a flashlight near by give that a try.

**MAKING IT BETTER****Reverse the response:**

Perhaps you would like the opposite response. Don't worry we can easily reverse this response just change:

```

analogWrite(ledPin, lightLevel); ---->
analogWrite(ledPin, 255 - lightLevel);

```

Upload and watch the response change:

**Night light:**

Rather than controlling the brightness of the LED in response to light, let's instead turn it on or off based on a threshold value. Change the loop() code with.

```

void loop(){
  int threshold = 300;
  if(analogRead(lightPin) > threshold){
    digitalWrite(ledPin, HIGH);
  }else{
    digitalWrite(ledPin, LOW);
  }
}

```

**Light controlled servo:**

Let's use our newly found light sensing skills to control a servo (and at the same time engage in a little bit of Arduino code hacking). Wire up a servo connected to pin 9 (like in CIRC-04). Then open the Knob example program (the same one we used in CIRC-08) **File > Examples > Servo > Knob**. Upload the code to your board and watch as it works unmodified.

**Using the full range of your servo:**

You'll notice that the servo will only operate over a limited portion of its range. This is because with the voltage dividing circuit we use the voltage on analog pin 0 will not range from 0 to 5 volts but instead between two lesser values (these values will change based on your setup). To fix this play with the val = map(val, 0, 1023, 0, 179); line. For hints on what to do visit <http://arduino.cc/en/Reference/Map>.

**MORE, MORE, MORE:**

More details, where to buy more parts, where to ask more questions:

**<http://ardx.org/CIRC09>**